Technical Report - Spam Detection

Lennart Kasserra

Introduction

An introduction should introduce the problem and data you're working on, give some background and relevant detail for the reader, and explain why it is important.

Analysis

Any exploratory analysis of your data, and general summary of the data (e.g. summary statistics, correlation heatmaps, graphs, information about the data…). Tell the reader about the types of variables you have and some general information about them, Plots and/or Tables are always great. This should also include any cleaning and joining you did.

Methods

The main model is a deep neural network with four hidden layers. Below is a brief technical overview of the model's architecture and its parameters, as well as the hyperparameters I used.

Learning rate scheduling was set up to reduce the learning rate by a factor of 0.8 after three epochs of patience[1] ("Reduce on Pleateau"). Early stopping would kick in after validation loss has not decreased for five epochs in a row. Binary cross-entropy loss was used as the loss function for training.

To select a proper competitor to compare the deep neural network against, I evaluated three "traditional" models: a penalized logistic regression model, a gaussian naive bayes classifier, and a random forest. For all models, I tuned their hyperparameters using regular grid search

[1] I.e. after validation performance did not increase for three successive epochs. This is meant to stabilize training & help the model settle into minima.

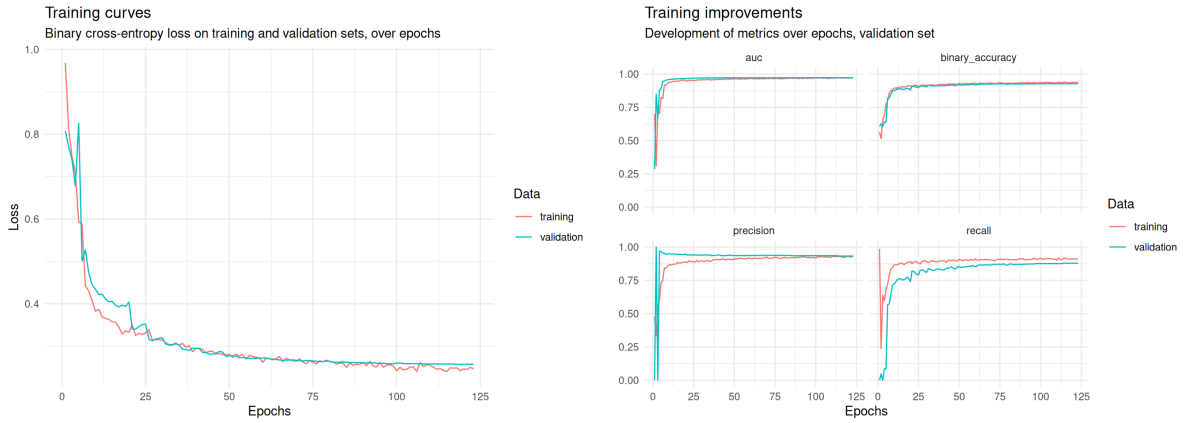
Layer type	Shape (units)	Activation
Input layer	n Input features	*None*
Dense (hidden)	128	ReLU
Dropout	*128*	*None*
Dense (hidden)	64	ReLU
Dropout	*64*	*None*
Dense (hidden)	32	ReLU
Dropout	*32*	*None*
Dense (hidden)	16	ReLU
Dropout	*16*	*None*
Output layer	1	Sigmoid

Total parameters: 43,397

Figure 1: Architecture

Hyperparameter	Value
Learning rate	0.001 (scheduled)
Optimizer	Adam
Dropout Rate	0.25
Batch size	32
Epochs	max. 250 (early stopping)
L2 Regularization	0.001

Figure 2: Hyperparameters



(a) Loss curves

(b) Training curves for different metrics

Figure 3: Training process of the deep neural network.

with ten-fold cross validation [2], and then compared their performance on the training set to that of the neural network. Results can be found in FIGURE X.

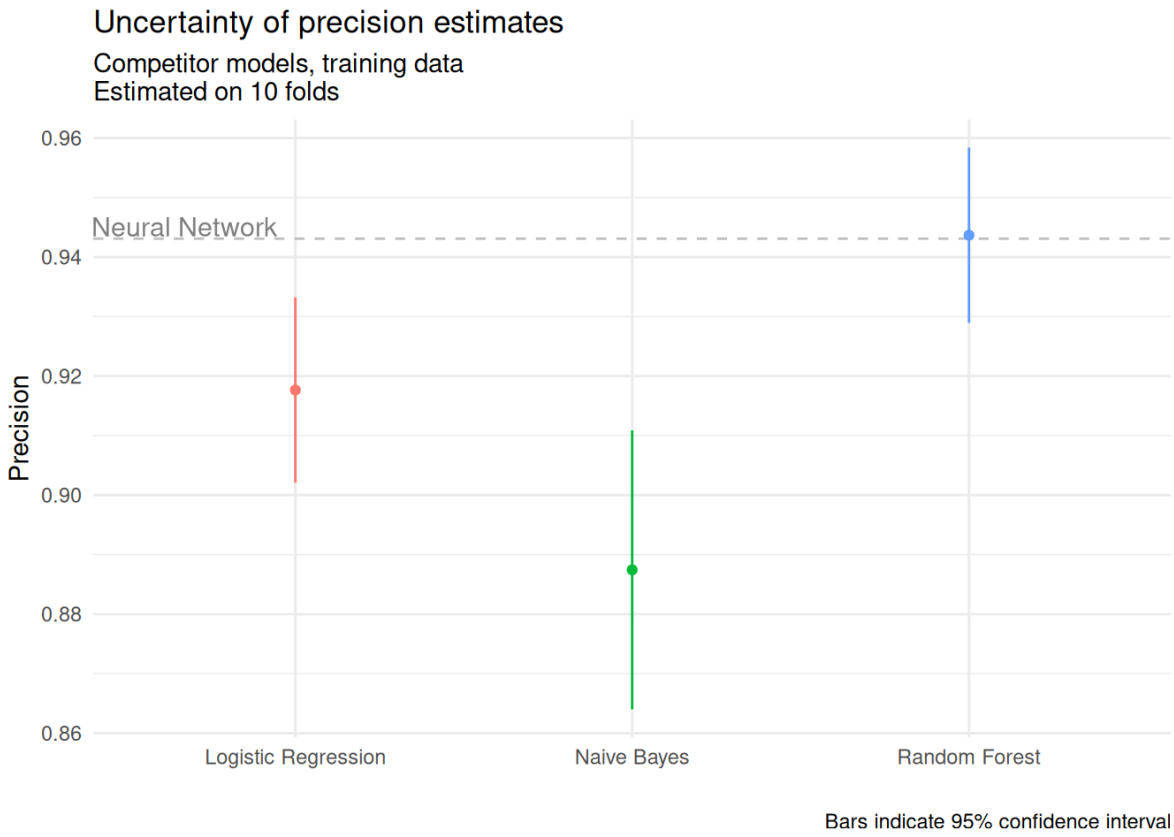


Figure 4: Potential competitors versus the deep neural network, evaluated on the training set. The 95% confidence intervals were calculated across the ten folds used for tuning.

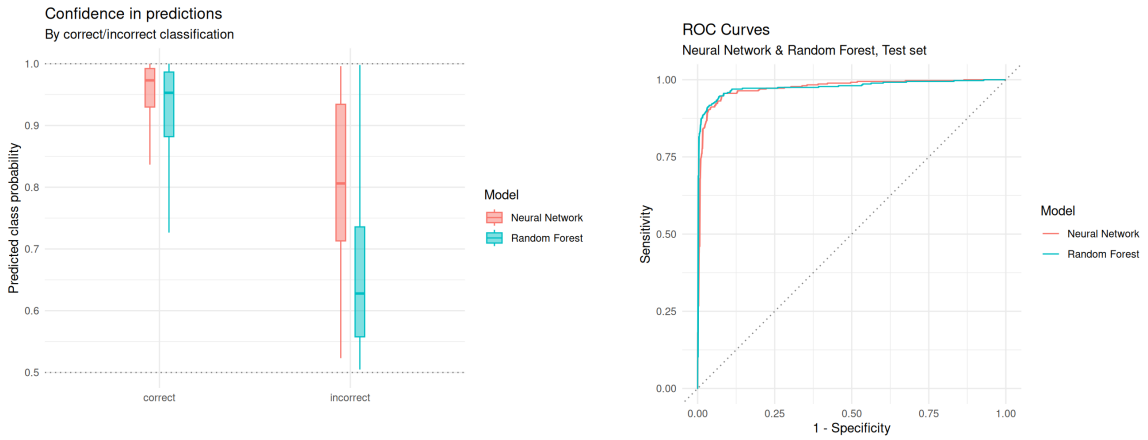
Results

The deep neural network was outperformed by a tuned Random Forest model on all metrics when evaluated on the test set. Crucially, it delivered lower precision than the Random Forest, but also fell behind on accuracy and f1-score. While the performance differences are at best marginal, when combined with the explainability that Random Forest offers over a deep neural network this is a solid case for using a "traditional" over a deep learning approach. Below are the metrics computed on the test set.

[2]Except for the gaussian naive bayes, where I did not tune the smoothing parameter & just used raw probabilities instead (the smoothing should not have any effect in this setting).

Model	Precision	Accuracy	F1
Neural Network	0.932	0.939	0.922
Random Forest	**0.935**	**0.942**	**0.926**

Additionally, I examined the models' "confidence" in their classifications by comparing their outputted class probabilities when being correct versus when being wrong. On average, the neural network is also more confident in misclassifications, which might hint at a deeper structural problem. However, examining the ROC curves shows just how close together the two models are in terms of performance.



(a) Model confidence: predicted class probabilities for right and wrong classifications.

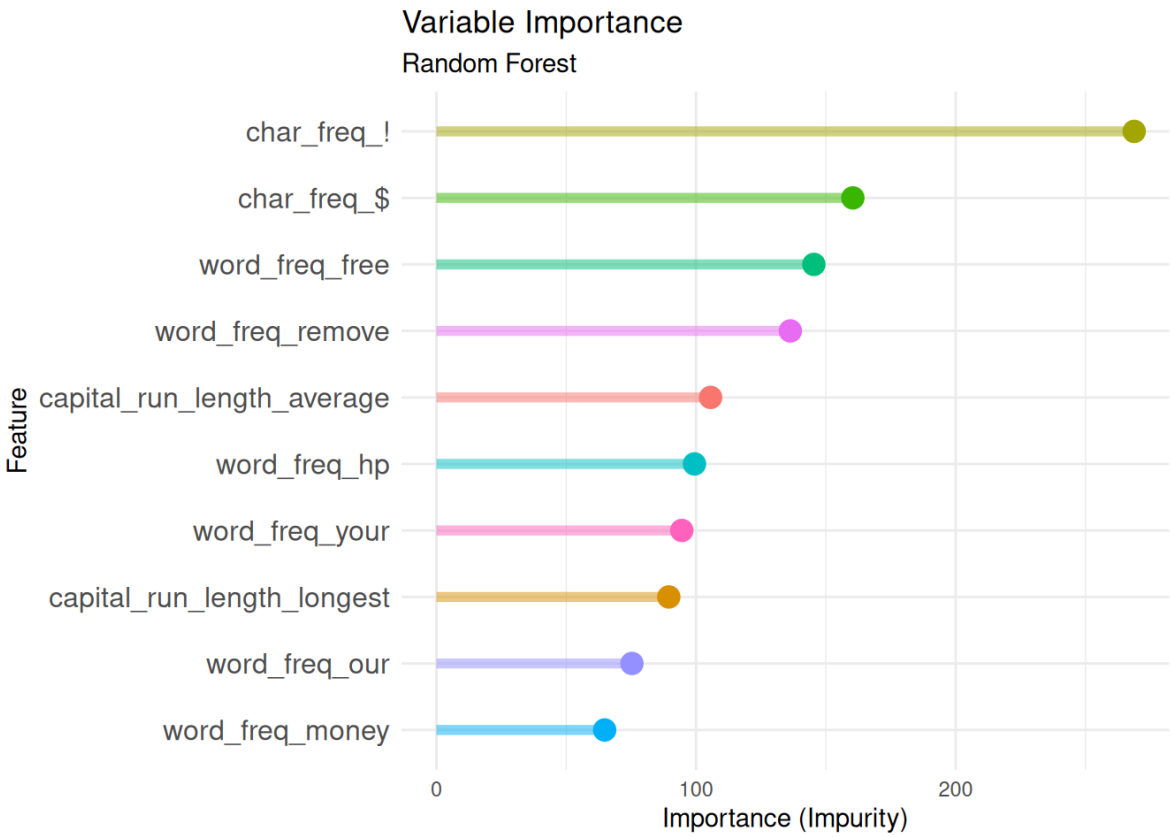
(b) Receiver-operating characteristic (ROC) curves.

Figure 5: Confidence & Test-set ROC curves.

Over all, a deep learning approach is not neccessary for this use case, and given this data.

Reflection

Given we decided on the Random Forest model, we can try to generate some explicit knowledge about our data. We can do this by examining variable importance, which I kept track of when building the model.



It seems that the number of exclamation marks and dollar signs, as well as promising free stuff are a giveaway about the true label of an observation. This is perhaps not a groundbreaking insight, but nevertheless illustrates once more the value of "traditional" models vis-a-vis deep learning approaches, where generating explicit knowledge about the data and classes is much less straight-forward, if at all possible.

Regarding possible improvements, it would have been tempting to try out more "traditional" models, and to explore a larger search space when tuning them, as it seems that when properly tuned they may outperform deep learning models on this task.

It might also have been fruitful to explore how smaller or more shallow neural networks compare to the deep model.